# 第２章　分散並列型動的解析手法

## 2.1 はじめに

本章では、分散並列型動的解析手法、並びに同システムの評価について述べる。SPACE で使用されている反復解法が並列処理向きであることから、基本的には分散並列型動的解析手法として、同じ反復解法が使用される。また、部材の弾塑性チェックと接線剛性行列の計算が全処理中で最も時間のかかる処理であることが、SPACE を用いて解析した結果より分析されている。そこで、分散並列型動的解析システムでは、この解析時間のかかる部分を並列化し、計算の効率化を目指す。

最初に、この反復解法について説明し、さらにどのように並列処理を行っているかについて解説する。分散並列型動的解析システムでは、主たる動的解析を行う計算機をマスターと呼び、他の計算機をスレーブと呼ぶ。このマスターとスレーブで構成される PC クラスターを用いて、動的解析が実行されることになる。Windows のソケット技術を元に PC クラスターを構築し、動的解析時では LAN による通信システムを用いて情報の共有化を図る。現在は、このシステムを用いて各種の数値実験や安定性などを調査する段階であり、未だ、このシステムを公開するまでには至っていない。ここでは、現在までの開発過程とシステムの特徴、その評価を示すことにする。

## 2.2 分散並列型動的解析手法

### 2.2.1 反復解法の基礎式

構造物の動的解析では、数値解析手法としてニューマークβ法が多用されている。同法はパラメータを適切に設定すれば、安定した解が得られ、特に、β= 1/4 で無条件安定となることが知られている。一方、大規模行列による数値解析では多量の演算が必要となり、特に、膨大な解析ステップを実行する動的解析では、この部分の効率化が重要となる。そこで、非線形振動方程式を変更して、各増分ステップで反復解法となる手法を提案し、同法が動的解析に対し効果的であること、また分散並列処理向きであることを示す。

一般に、構造物の非線型振動方程式は次式のように表される。

$$[M]\{\ddot{y}\}+[C]\{\dot{y}\}+\{K(y)\}=-[M][I]\{\ddot{u}_g\}+\{P_S\} \qquad \cdots\cdots\cdots\cdots(2.1)$$

ここで、左辺第１項は慣性力、第２項は減衰項、第３項は幾何学的及び材料非線形性を有する構造物の内力ベクトルである。ベクトル$\{\ddot{y}\},\{\dot{y}\},\{y\}$は、加速度、速度、変位を各々表す。また、右辺第１項は地震による外力、第

2 項は構造物に直接加わる荷重であり、時間に依存すれば風荷重に、依存しなければ擬似的な静的荷重に対応する。地震加速度ベクトル$\{\ddot{u}_g\}$は各方向ｘ,ｙ,ｚに関する加速度$(\ddot{u}_g, \ddot{v}_g, \ddot{w}_g)$である。行列$[I]$は、各方向の変位に対応する位置で１に、他は0となるベクトル$\{I_x\},\{I_y\},\{I_z\}$から構成される。

$$\left.\begin{aligned}\{\ddot{u}_g\} &= (\ddot{u}_g, \ddot{v}_g, \ddot{w}_g)\\ [I] &= \left[\{I_x\},\{I_y\},\{I_z\}\right]\end{aligned}\right\} \cdots\cdots\cdots\cdots (2.2)$$

最初に、式(2.1)中の内力ベクトルの非線形部分を右辺に移項し、左辺には線形項のみ残して、振動方程式を以下のように変更する。

$$\left.\begin{aligned}[M]\{\ddot{y}\}+[C]\{\dot{y}\}+[K]\{y\} =&\\ -[M][I]\{\ddot{u}_g\}+\{P_S\}-\{K(y)\}+[K]\{y\}&\end{aligned}\right\} \cdots\cdots\cdots\cdots (2.3)$$

さらに、右辺の非線形内力項を増分変位が小さいとして、以下のように書き換える。

$$\{K(y)\} = \{K(\bar{y})\}+[K_T(\bar{y})]\{\Delta y\} = \{Q(\bar{y})\}+[K_T(\bar{y})]\{\Delta y\} \qquad \cdots\cdots\cdots\cdots (2.4)$$

ここで、$\bar{y}$は増分時間前の変位であり、$\Delta y$は増分変位である。また、$\{Q(\bar{y})\}$は部材応力から求めた節点力ベクトルを、$[K_T(\bar{y})]$は増分前の接線剛性を各々示す。右辺の非線形内力項を節点力ベクトルで表すことから、増分前までの不釣合い力が増分後の右辺項で評価され、方程式を解いた後、増分前の誤差が打ち消され、誤差の増大を防いでいる。上式を式(2.3)に代入すると、振動方程式は下式のように変更され、反復解法の基礎式が得られる。

$$\begin{aligned}[M]\{\ddot{y}\}+[C]\{\dot{y}\}+[K]\{y\} = &-[M][I]\{\ddot{u}_g\}\\ &+\{P_S\}-\{Q(\bar{y})\}-[K_T(\bar{y})]\{\Delta y\}+[K]\{y\}\end{aligned} \cdots\cdots\cdots\cdots (2.5)$$

上式にニューマークβ法を適用する。まず、増分時間Δｔ秒後の変位ベクトル$\{y_{n+1}\}$と速度ベクトル$\{\dot{y}_{n+1}\}$を、

$$\left.\begin{aligned}\{\dot{y}_{n+1}\} &= \{\dot{y}_n\}+\{(1-\delta)\{\ddot{y}_n\}+\delta\{\ddot{y}_{n+1}\}\}\Delta t\\ \{\Delta y_{n+1}\} &= \{\dot{y}_n\}\Delta t+\{(0.5-\beta)\{\ddot{y}_n\}+\beta\{\ddot{y}_{n+1}\}\}\Delta t^2\\ \{y_{n+1}\} &= \{y_n\}+\{\Delta y_n\}\end{aligned}\right\} \cdots\cdots\cdots\cdots (2.6)$$

で仮定する。ここで、添え字のｎは増分前の値、n+1は増分後の値、$\delta,\beta$は数値計算用パラメータを表す。次に、上式中の係数を

$$\left.\begin{aligned}\{a\} &= \{\dot{y}_n\}+\Delta t(1-\delta)\{\ddot{y}_n\}\\ \{b\} &= \Delta t\{\dot{y}_n\}+\Delta t^2(0.5-\beta)\{\ddot{y}_n\}\end{aligned}\right\} \cdots\cdots\cdots\cdots (2.7)$$

$$\left.\begin{aligned}\{\bar{b}\} &= \{y_n\} + \{b\} \\ \mu_1 &= \Delta t\delta \\ \mu_2 &= \Delta t^2\beta\end{aligned}\right\} \cdots\cdots\cdots\cdots (2.8)$$

としてまとめ、式(2.6)と(2.7)を反復解法の基礎式(2.5)に代入すると、増分後の加速度を未知ベクトルとする釣合式が以下のように得られる。

$$\begin{aligned}[M]\{\ddot{y}_{n+1}\} + [C]\{\{a\} + \mu_1\{\ddot{y}_{n+1}\}\} + [K]\left\{\{\bar{b}\} + \mu_2\{\ddot{y}_{n+1}\}\right\} \\ = -[M][I]\{\ddot{u}_g\} + \{P_S\} - \{Q(y_n)\} \\ -[K_T(y_n)]\{\{b\} + \mu_2\{\ddot{y}_{n+1}\}\} + [K]\left\{\{\bar{b}\} + \mu_2\{\ddot{y}_{n+1}\}\right\}\end{aligned} \quad \cdots\cdots (2.9)$$

さらに、上式を整理すると、

$$\begin{aligned}\left[[M] + \mu_1[C] + \mu_2[K]\right]\{\ddot{y}_{n+1}\} \\ = -[M][I]\{\ddot{u}_g\} + \{P_S\} - \{Q(y_n)\} - [C]\{a\} - [K_T(y_n)]\{b\} \\ + \mu_2\left([K] - [K_T(y_n)]\right)\{\ddot{y}_{n+1}\}\end{aligned} \quad \cdots\cdots\cdots\cdots (2.10)$$

となる。ここで、係数を以下のように

$$\left.\begin{aligned}[F] &= \left[[M] + \mu_1[C] + \mu_2[K]\right] \\ [G] &= \mu_2\left([K] - [K_T(y_n)]\right) \\ \{g\} &= -[M][I]\{\ddot{u}_g\} + \{P_S\} - \{Q(y_n)\} \\ &\quad -[C]\{a\} - [K_T(y_n)]\{b\}\end{aligned}\right\} \cdots\cdots\cdots\cdots (2.11)$$

とすると、方程式は、

$$[F]\{\ddot{y}_{n+1}\} = [G]\{\ddot{y}_{n+1}\} + \{g\} \quad \cdots\cdots\cdots\cdots (2.12)$$

となる。上式から、ガウス・ザイデル法などと同様、下に示す分離型の反復式が得られる。

$$\{\ddot{y}_{n+1}\}^{i+1} = [F]^{-1}[G]\{\ddot{y}_{n+1}\}^{i} + [F]^{-1}\{g\} \quad \cdots\cdots\cdots\cdots (2.13)$$

ここで、上添え字 $i$ は、反復回数を表す。

提案する並列処理システムでは、上に示す反復式を効率良く、また記憶領域の少量化を考慮して計算する方法が用いられている。例えば、式(2.11)中の係数行列$[F]$は、行列 3 つの足し算になっているが、行列を作成した後に求めるのでなく、直接に係数行列$[F]$に組み込む方式がとられている。特に、減衰行列$[C]$は、レーリー減衰を使用する場合、各部材の剛性行列や節点質量から直接係数行列$[F]$に組み込まれる。また、右辺項の行列$[G]$とベクトル$\{g\}$ は、共に行列を用いて評価されているが、最終的にはベク

トルとして得られれば良く、各行列は全体行列として計算する必要はない。ここでは、下式のように各部材ごとに部材行列を求め、対応する変位、あるいは速度と掛け算し、右辺項ベクトルに組み込むことになる。この方法を用いることによって、大きな記憶領域を必要としなくなる。ここでは、以下のように式(2.12)を変更して使用する。

$$\left.\begin{aligned} \{\ddot{y}_{n+1}\}^{i+1} &= [F]^{-1}\{z\}^{i} \\ \{z\}^{i} &= \sum_{i=1}^{m}\left([G_i]\{\ddot{y}_{n+1}\}^{i} + \{g_i\}\right) \end{aligned}\right\} \cdots\cdots\cdots\cdots (2.14)$$

ここで、ベクトル$\{z\}^{i}$は、部材毎の$[G_i]$と対応する加速度$\{\ddot{y}_{n+1}\}^{i}$を掛け算し、また部材の$\{g_i\}$を加えて、変位の適合を用いて重ね合わせたものである。並列処理では、各スレーブの担当部材毎に上記のベクトル$\{z\}^{i}$を計算し、マスター計算機に集めて総和をとった後、式(2.13)の方程式を解くことになる。

提案した反復解法を用いると、係数行列$[F]$は線形であるため、一度LDU分解すれば各ステップで用いることができ、解を求めるための計算時間はほとんど要しない。さらに、部材の塑性チェックや応力計算、接線剛性、あるいは反復解法の右辺項の計算等は部材相互に関連しないため、部材毎に独立して計算が可能であり、このことからも、ここで提案する反復解法は分散並列処理向きであると言える。

### 2.2.2 反復解法の初期値

反復式(2.12)における初期値は、Δｔ 秒前の値を用い、ニューマークβ法を利用して以下のように表す。

$$\left.\begin{aligned} \ddot{y}_{n+1} &= \ddot{y}_n + \dddot{y}_n \Delta t \\ \dot{y}_{n+1} &= \dot{y}_n + \ddot{y}_n \Delta t + \delta \dddot{y}_n \Delta t^2 \\ y_{n+1} &= y_n + \dot{y}_n \Delta t + 0.5\ddot{y}_n \Delta t^2 + \beta \dddot{y}_n \Delta t^3 \end{aligned}\right\} \cdots\cdots\cdots\cdots (2.15)$$

上式を初期値として用いると、Δｔ 秒前の変位の 3 回微分が必要となる。ここでは、2Δｔ秒前の加速度$\ddot{y}_{n-1}$を利用して下式で与える。

$$\dddot{y}_n = \frac{\ddot{y}_n - \ddot{y}_{n-1}}{\Delta t} \cdots\cdots\cdots\cdots (2.16)$$

上式を式(2.15)に代入すると、精度の良い反復式の初期値が次のように得られる。

$$\left.\begin{aligned} \dot{y}_{n+1} &= \dot{y}_n + \left\{(1+\delta)\ddot{y}_n - \delta \ddot{y}_{n-1}\right\}\Delta t \\ y_{n+1} &= y_n + \dot{y}_n \Delta t + \left\{(0.5+\beta)\ddot{y}_n - \beta \ddot{y}_{n-1}\right\}\Delta t^2 \end{aligned}\right\} \cdots\cdots\cdots\cdots (2.17)$$

### 2.2.3 反復解法における発散

反復解法では、式(2.13)より以下に示す反復行列$[T]$の特性によって、収束の速さや、また解が発散するか否かが決められる。

$$[T]=[F]^{-1}[G] \qquad \cdots\cdots\cdots\cdots(2.18)$$

反復行列$[T]$の変化は、式(2.11)を参考にすると、時間が進んでも行列$[F]$は一定であるため、行列$[G]$内の$[K_T(y_n)]$に依存する。例えば、変位が小さく、接線剛性行列$[K_T(y_n)]$が線形の剛性行列に等しいとき、行列$[G]$はゼロ行列となり、そのため、式(2.13)は反復式とならず、一回で解が求まることになる。

反復解法の収束条件は、反復行列のスペクトル半径$\rho(T)$によって以下のように表される[17)]。

$$\rho(T) = \max|\lambda_i| < 1 \qquad \cdots\cdots\cdots\cdots(2.19)$$

ここで、$\lambda_i$は反復行列$[T]$の $i$ 番目の固有値である。一般にスペクトル半径を直接求めるには時間がかかり、増分ステップ毎に求めることは当然できない。そこで、以下に示すスペクトル半径の上界を用いて、反復行列の特徴を分析する。反復行列の上界は以下のように表すことができる。

$$\rho(T) \le \max\sum_{j=1}^{n}|T_{ij}| = \|T\| \qquad \cdots\cdots\cdots\cdots(2.20)$$

ここで、$\|T\|$は行列$[T]$のノルムを示す。行列に関するノルムの性質と式(2.19)、(2.20)から、スペクトル半径の上限は、

$$\rho(T) \le \left\|[F]^{-1}[G]\right\| \le \frac{\|G\|}{\|F\|} = \frac{\|K-K_T\|}{\|F\|} \qquad \cdots\cdots\cdots\cdots(2.21)$$

で得られる。

動的解析中における反復行列のノルムの変化は、行列$[G]$によって生じる。前述したように、応答が線形振動状態の場合は、行列$[G]$はゼロ行列となり、スペクトル半径はゼロで、式(2.12)は反復式でなくなる。一方、非線形性が強くなると接線剛性行列のノルム$\|K_T\|$は一般的に小さくなり、その結果、式(2.21)の分子は大きくなる。さらに、構造物が座屈やメカニズムなど不安定な状態に至るとき、接線剛性行列の固有値のひとつがゼロや負となり、反復行列のスペクトル半径は１に近づいて収束が遅くなることや、場合によっては１を超えて解が発散することもある。

式(2.21)のノルム$\|F\|$は、解析途中で増分時間Δｔを変化させなければ解析中変わらない。初期設定でΔｔをより小さくすると、Δｔの二乗に比例す

る剛性項に比較して、一乗に比例する減衰項や定数である質量項が行列$[F]$で優位となる。また、行列$[G]$もΔt の二乗に比例するため、Δt を小さくすることが、ノルム$\|F\|$を相対的にノルム$\|G\|$より大きくさせ、スペクトル半径を小さくさせることになる。その結果、解の発散を防ぎ、収束範囲を拡大させる。

ここで示した反復解法は、系の非線形性が弱い場合は、収束も早く、陰解法に比較すると有利な点が多い。しかし、系の非線形性が強い場合、特に動座屈や塑性によるメカニズム等を扱う場合は、解が発散する可能性があることを考慮して、次善の策を用意しておく必要がある。例えば、陰解法に切り替える方法、あるいは、解析途中で増分時間を小さくする方法などがあろう。無論、次善の方法で、発散する部分を通りすぎると元の反復解法に戻ることになる。前者は、単一のコンピュータでは、システムの構成上組み込み易くしかも効果的である。しかし、ここで提案している分散並列処理に陰解法を用いると、接線剛性など多くの情報を PC 間で転送する必要が生じ、それに多くの時間を取られるため並列処理の有効性が相殺される。このため、陰解法を併用する手法は、提案している分散並列処理には適当でないと言える。

一方、後者は、解析途中で増分時間を変化させることによって達成され、増分時間の変化による各種の影響を合理的に処理することが可能であるならば、計算方法が変化するわけではないので、分散並列処理向きと言える。ただし、どの程度Δt を小さくすべきか、また、変更後発散するか否かが計算前に明らかではない。これらに対処した上で、分散並列処理システムに同法を組み込む必要があるが、これについては今後の検討課題としよう。

## 2.3 動的解析システムの検討

### 2.3.1 単一動的解析システムの分析

ここでは、反復解法を基本にした動的解析システムを分析し、並列処理が可能となる部分を抽出する。まず、SPACE を用いて標準的な構造物の応答解析を行い、分析した結果を図 2-1 に示す。解析モデルは、ファイバーモデルを用いた単層ラチスドールで、自由度約 2000、Δt = 0.001 秒、解析時間 T=10 秒である。なお、ここで用いている動的解析システムの数値解析手順が図 2-2 に示されている。

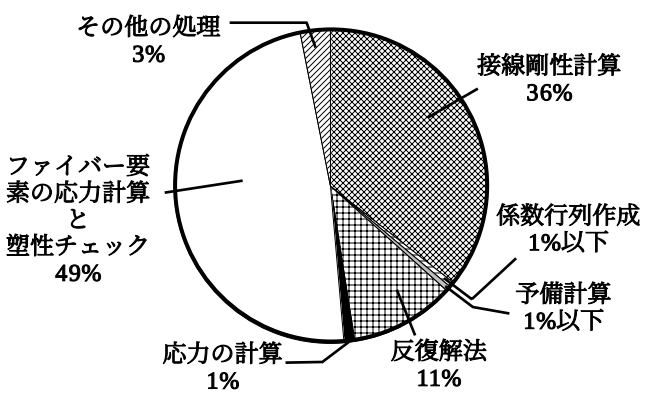


**図 2-1　標準的な処理時間**

図 2-1 で理解できるように、

解析に要した時間の大半はファイバー要素の応力計算、塑性チェックと接線剛性の計算で、全体に対し約 85% となっている。さらに、ニューマークβ法で行う反復解法の部分を含むと全体の解析時間のなんと 96% となる。この４つの処理は図 2-3 の解析フロー中の網掛け部分に示されている。ここでの処理は部材毎に独立であり、この４つの処理の大半は並列処理向きであるといえる。

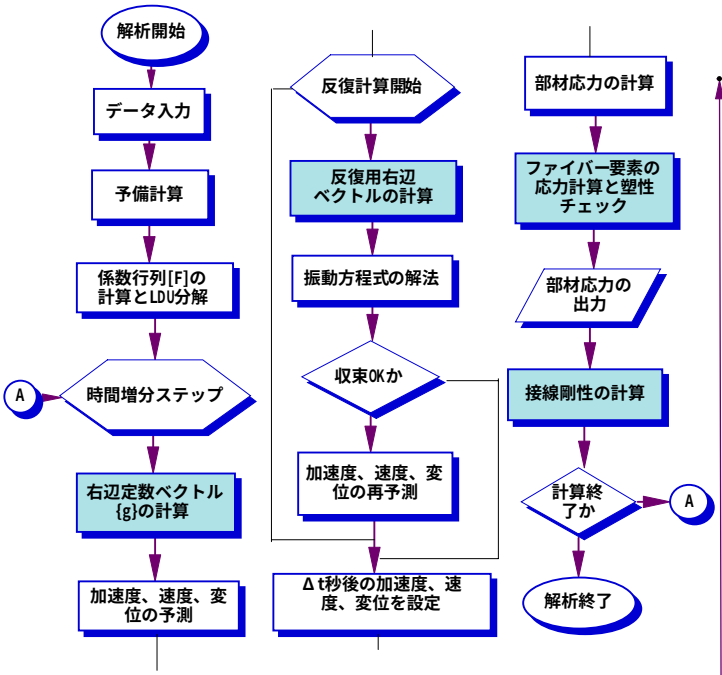


**図 2-2　単一応答処理のフローチャート**

例えば、ファイバー要素の応力計算と接線剛性の計算は、部材相互に関連しないため、解析前に各スレーブに担当部材の範囲を通知し、その情報に従いスレーブ側で行う。計算された情報を一括してマスターが受け取り、データを統合して解析が進められる。ここでは、各部材の情報は担当するスレーブ側で持つことになり、マスター側では必ずしも全てを必要とせず、ファイバーデータなどのワーク領域を大幅に減らすことができる。ただし、並列化に当たって問題となるのは、マスター側とスレーブ側で如何に情報を共有するかということであり、これについては後節で説明する。

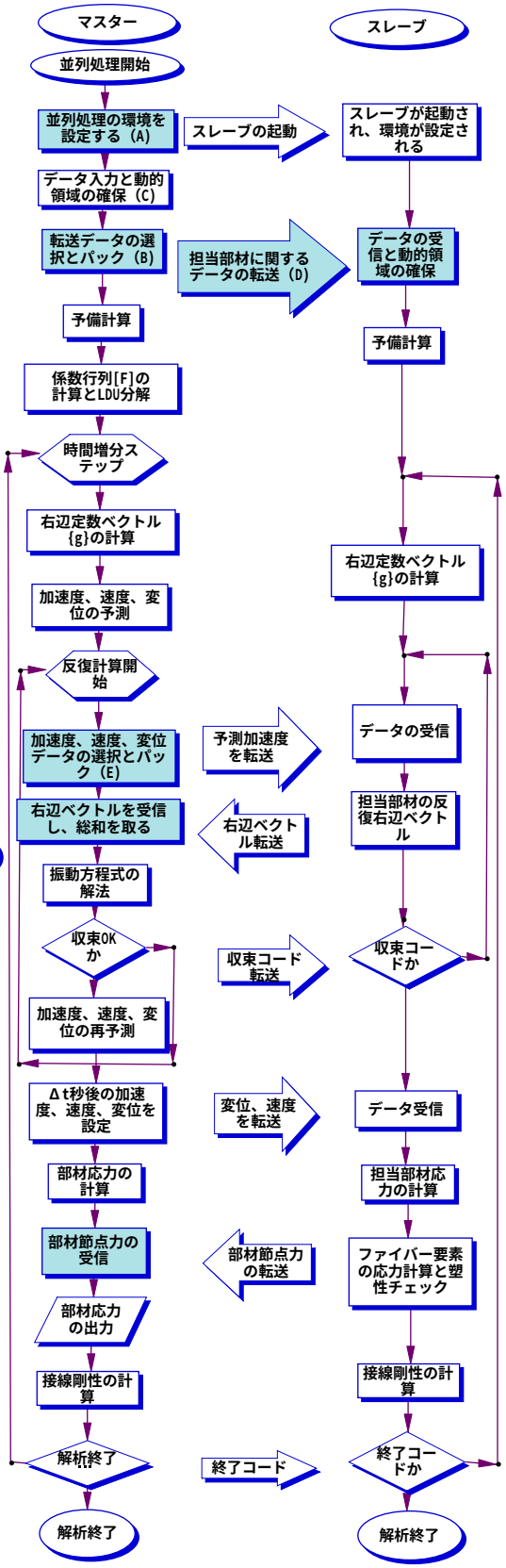


**図 2-3 並列処理におけるデータ送受信**

### 2.3.2 動的解析システムの並列処理化

上述した動的解析システムの分析を元に、図 2-2 に示す解析フローを分散並列処理用に変更し、図 2-3 に示す。分散並列システムは、一つのマスターと多数のスレーブとで構成されており、各々処理を分担する。ただし、マスターは担当部材を持つことも可能であり、スレーブと同様の計算を行うこともできる。

SPACE で採用する分散並列型処理システムの動的解析手法は、基本的には単一処理と何ら変わるところはない。ただ上記 4 つの処理を各スレーブ側で分担するため、各種のデータを転送する必要が生じる。転送システムを構築するために、単一処理にはない並列処理特有のプロセスが必要となる。それらを以下にまとめ、その特徴を述べる。

1）PC 間のネットワークを構築するための環境を設定する。（図 2-3 の A 部分）
2）マスター側で計算コントロールデータや各種のデータを入力し、スレーブ側に転送する。（図 2-3 の B 部分：入力データの一元管理）
3）スレーブ側における分担部材に対応する記憶領域の動的確保と釣合式右辺項の計算を行う。（図 2-3 の C 部分：スレーブ側の動的領域確保）
4）各種データの送受信システムを構築する。（図 2-3 の D 部分：データの転送システム）
5）各スレーブが担当する部材と節点に関するデータを分類し、必要とするデータにまとめる、逆にまとめたデータを元の全節点に関するデータに復帰する。（図 2-3 の E 部分）

現在のシステムは、さらにスレーブ制御コードの送信部分に改良を加え、Ver.1.11 となっている

上記の並列処理特有の処理は、後章で詳細に検討する。開発当初は、公開されていた Windows PC 用の WMPI を用いていたが、開発を進めていく中で、不安定性が目立つようになってきた。そこで、開発プロジェクトチームは、MPI に準拠したライブラリーを開発し、動的解析システムに組み込んだ。その結果、分散並列型動的解析システム Ver.1.10 となったわけである。後章では、このライブラリーについて詳細に説明する。

## 2.4 分散並列型動的解析システムの構築

### 2.4.1 システムの概要

最初に、改良前のシステムとその評価について説明する。SPACE は、誰もが容易に使用できる計算機環境を考慮した実用的な構造解析システムとなることを目指している。そこで、分散並列型動的解析システムでも、一般的な PC による並列処理用クラスターを構築し、仮想的な分

散メモリ型の並列計算機システムを対象にする。システムの有効性を検討するために構築した PC クラスターのプロトタイプを図 2-4 に、その仕様を表 2-1 に示す。ここでは、システムを管理する PC をマスターとし、分担して計算を請負う PC をスレーブとする。また、各 PC は、100BASE-TX イーサネットを用いてスイッチングハブで接続される。この PC クラスターは、LAN に接続することにより遠隔利用を可能としている。

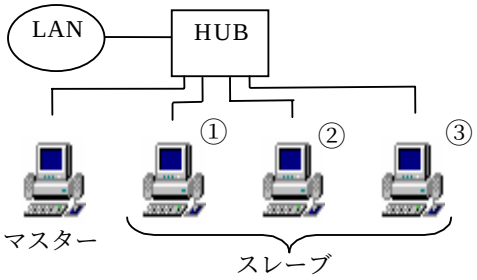


**図 2-4 PC クラスターの構成**

**表 2-1　PC クラスターの仕様**

| | |
|---|---|
| PC の仕様 | |
| マスター | Pentium Ⅱ 450MHz メモリ 386MB |
| スレーブ①②③ | Pentium Ⅱ 450MHz メモリ 256MB |
| ネットワーク仕様 | |
| ハブ | 100Mbps スイッチングハブ |
| ケーブル | カテゴリ５、ツイストペアケーブル |
| ネットワークカード | 100BASE-TX イーサネット |

この分散並列型処理システムの特徴は、Windows98/NT/2000/XP（以下Windowsと略記）が動作している一般的なPC上に、並列処理用のソフトウエアを起動することで、並列処理用PCクラスターを構築している点である。他に見られる多くのPCクラスターは、コストや性能を優先してUNIXの利用が一般的であるが、SPACEでは汎用面に重点をおき、GUI（Graphic User Interface）を含めて、Windows上で動作する分散並列型構造解析システムを構築している。

## 2.4.2 システムの構成

図2-5は、Windowsと並列化構造解析システムとの関係を示す。並列処理を実現するため、WMPIライブラリーが利用され、また通信を管理するため、常駐するプログラムとしてWMPIデーモンが起動される。WMPIデーモンはマスターとスレーブ間の送受信を管理し、実行する。実際の通信処理はWMPIが提供するライブラリーを用いて行うため、TCP/IPに関する複雑な設定を行う必要がない。

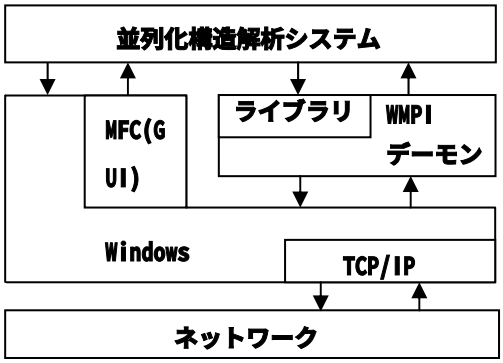


**図 2-5 並列化構造解析システムの動作環境**

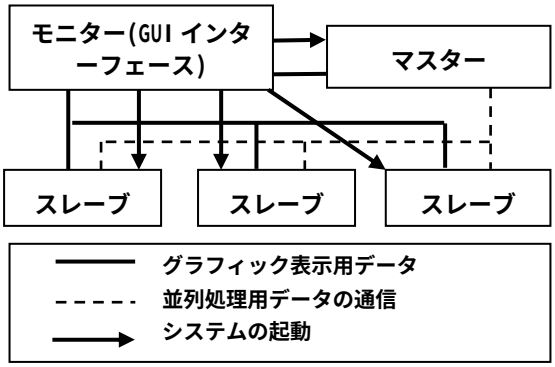


**図 2-6 並列化構造解析システムの構成**

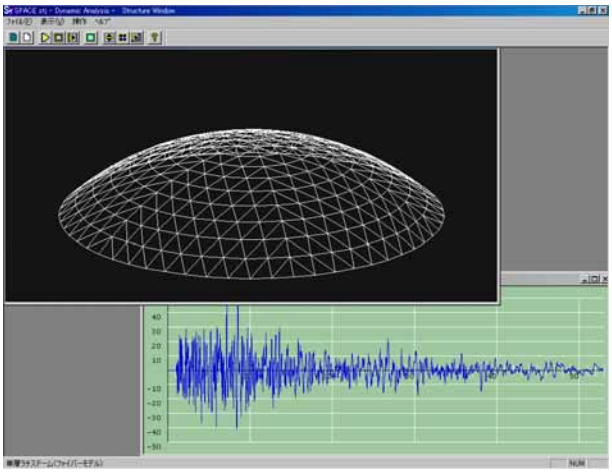

**図 2-7 ウインドウシステムによる GUI**

図 2-6 は、分散並列型構造解析システムにおけるモニター、マスター、スレーブの関係とネットワークによるデータ送受信の関係を表す。並列処理システムは、モニターが最初に立ち上がり、その後、モニターによって、マスターやスレーブが起動させられる。モニター（GUI インターフェイスを有する）の役割は、他に、解析途中でマスターやスレーブと通信を行い、解析経過をユーザーが理解し易いようにグラフィック等を用いて情報を提供する。図 2-7 はモニターの様子を示すものであり、マルチウインドウ形式で、三次元表示された変形図や節点変位を時刻歴で表示する。

プロトタイプで使用した通信方法は、マスターが任意のスレーブと通信する時、他のスレーブが待たされ、その処理が終了した後、次のスレーブとデータ交換が行われるという最も単純な方式を採用している。この方式では通信処理がマスターに集中し、スレーブに多くの待ち状態が発生することが予想される。これは今後の検討課題でもある。

## 2.5 分散並列システムに関する処理能力の分析

### 2.5.1 解析モデルの概要

解析モデル(部材は図2-8に示すファイバー要素を含むモデル)は円形平面を有する剛接合単層ラチスドームとし、その形状と骨組が図 2-9 に示される。モデルの規模による並列処理効率への影響を分析するため、各モデルの自由度はおおよそ 500, 1000, 2000, 4000 に設定し、各々その名前を D-500、D-1000、D-2000、D-4000 とする。設定した解析モデルの仕様を表 2-2 に示す。並列処理を実行するにあたって、担当する部材数はマスターとスレーブ共に同じ割合とし、CPU(Central Processing Unit)数で等しく分配する。なお、解析は動的応答解析を 0.001 秒刻みで 5 秒間行ったものであり、解析ステップは 5000 回である。

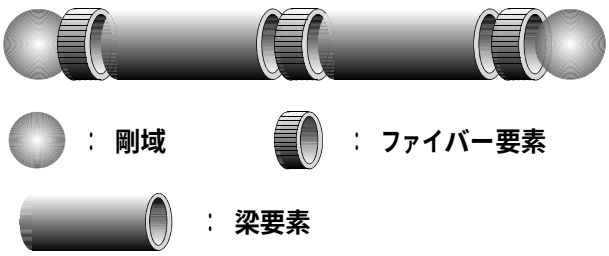


**図 2-8　部材モデル**

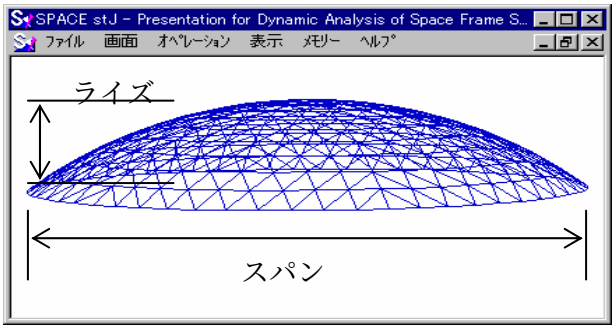


**図 2-9　解析モデルの形状**

### 2.5.2 並列処理の効果と通信時間の増加

ここでは、PC クラスターの大きさによって、提案した並列処理システムの並列計算効率がどのように変化するか、また、通信量の変化がどのように解析時間全体に影響を及ぼすのか、を分析する。PC クラスターとして、１から 4 までの CPU の数に対応する 4 つのモデルを設定する。

PC クラスターのシステム構成は、CPU 数が１の場合は、モニターとマスターが、同一 CPU 上に各１つのプロセスとして起動される。この場合、プロセス間のデータ通信は WMPI の仕様によりメモリ内で行われる。CPU 数が 2 以上の場合は、マスターとモニターが一つの PC 上に各1プロセスとして起動され、また、スレーブは、他の PC に１プロセスづつ起動される。複数の CPU で構成される場合、マスターとスレーブ間のデータ通信はネットワーク経由で行われる。

**表 2-2 解析モデルの諸元**

| モデル | スパン | ライズ | 部材数 | 節点数 | 自由度数 |
|---|---|---|---|---|---|
| D-500 | 100 | 4.4 | 240 | 91 | 456 |
| D-1000 | 140 | 17.5 | 462 | 169 | 888 |
| D-2000 | 200 | 36.4 | 960 | 331 | 1808 |
| D-4000 | 300 | 86.6 | 2070 | 721 | 4056 |

（スパンとライズの単位はm）

**表 2-3　並列処理における各処理の解析時間（秒）**

| | D-1000 | | | |
|---|---|---|---|---|
| プロセッサ数 | 1 | 2 | 3 | 4 |
| 予備計算 | 0.43 | 0.33 | 0.55 | 0.66 |
| 係数行列作成 | 0.10 | 0.11 | 0.23 | 0.16 |
| 反復解法 | 147.82 | 151.66 | 183.61 | 184.30 |
| 応力計算 | 34.52 | 56.88 | 38.78 | 41.38 |
| **ファイバー応力計算** | **425.38** | **354.83** | **251.38** | **178.45** |
| **接線剛性計算** | **349.84** | **235.46** | **178.25** | **114.52** |
| その他の処理 | 322.04 | 269.69 | 321.55 | 353.83 |
| 総解析時間 | 1220.22 | 897.38 | 765.26 | 646.79 |

並列処理を行うことによって、各処理時間がどのように短縮されるかについて、解析モデル D-1000 を用いて分析し、その結果を表 2-3 に示す。同表より、並列処理を導入したファイバー応力の計算と接線剛性行列の作成は、CPU 数の増加に対応して計算時間が短縮されていることが分かる。例えば、CPU 数を 1 から 4 にした場合で、ファイバー応力の計算にかかる時間は 42%に、接線剛性の計算は 32%となっている。ただし、今回の並列化のアルゴリズムではデータ通信量が比較的多いため、総解析時間は CPU の数が多くなるに従って、低減率の鈍化が見られる。一層の効率化を図るためには、更なるデータ通信の性能向上が期待されるところである。

### 2.5.3 自由度数の違いによる並列処理の効果

図2-10では、単一CPUによる陰解法と反復解法の解析時間を比較しており、反復解法の優位性を示している。 陰解法は自由度数の2から3乗に比例して計算時間を必要とするが、反復解法ではほぼ自由度数に比例する。図2-11 は、分散並列処理を行った結果であり、4つの解析モデルに対する総解析時間を示す。縦軸は総解析時間（秒）であり、横軸は自由度の数を示す。図中の印（○、□、◇、×）は、CPU数1,2,3,4に各々対応する。反復解法による並列計算の能力は、解析時間の増加傾向が単一CPU

の場合と同様で、陰解法と比較して、解析能力の優秀性を実証している。解析モデルの規模が変化しても、解析時間の増加率は自由度数にほぼ比例して得られており、極端に増えることはない。さらに構造物の規模が大きくなり、自由度が増せば、提案した並列構造解析法は、反復解法の有利さと並列処理による計算速度向上が期待され、一層の優位性が発揮されることになる。

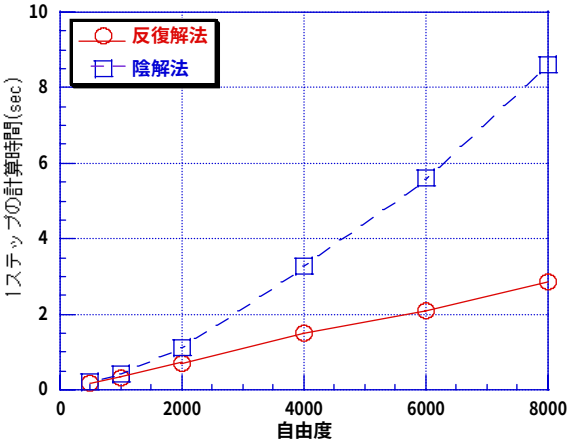


**図 2-10　反復解法と陰解法の処理能力の比較**

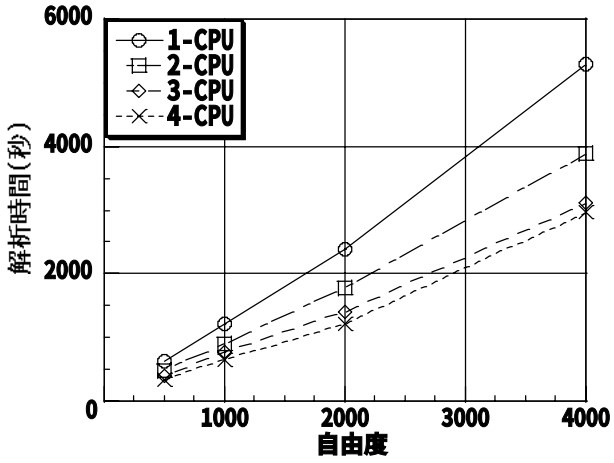


**図 2-11　自由度数と解析時間との関係**

### 2.5.4 CPU 数の違いによる並列処理の効果

図 2-12 は、並列処理による解析時間短縮の割合を示す。同図で、横軸は CPU 数を示し、縦軸は各解析モデルの１CPU で解析した時間に対する低減率を示す。並列度の増加に伴って（CPU 数の増加）、解析時間が減少していること、また、その傾向は自由度数の違いによらず一様であることが分かる。開発した分散並列型構造解析システムにおける解析時間の低減率は、各モデルの平均的な値として、CPU 数が 2 つで約 75%、3 つで 62%、4 つで 54%程度となっている。

本システムで並列処理を導入した部分が全体の 95%（図 2-1 参照）を占めるとし、また CPU 数が 4 の場合、解析時間の効率化の限界を理論的に示す Amdahl の法則によれば[16]、低減率は 36%となる（図 2-12 の理論値参照）。ここでの結果は 54%程度であり、比較的単純な並列計算アルゴリズムを利用しているにも関わらず効率化がかなり達成されていることが分かる。

ただし、CPU 数に対する解析時間の低減率は、CPU 数が 4 程度になると、並列処理の効果が少し

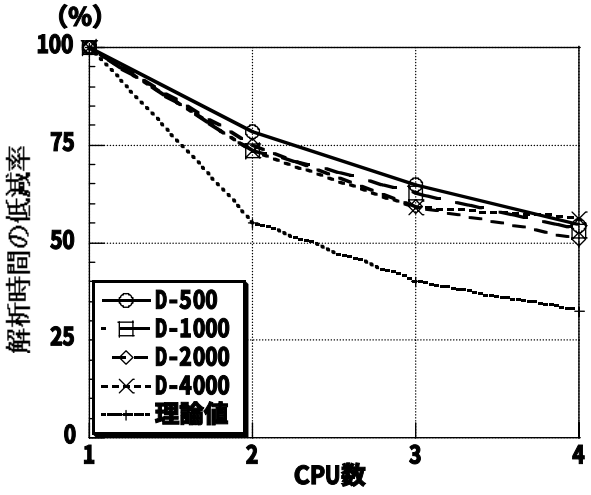


**図 2-12　CPU 数と解析時間の関係**

緩やかとなり、鈍化する傾向が見られる。特に CPU 数が 3 から 4 に増加した場合、解析時間の低減する割合はそれほど大きくなっていない。このことからも CPU 数が更に多くなると並列処理の効果があまり期待できないことを示している。その原因は、CPU 数の増加によって、CPU 間のデータ通信量が増加し、並列処理の効果を相殺してしまうからである。今後、多量の情報を効率的に送受信可能とする PC クラスターを構築する必要があろう。

## 2.6 並列処理における改良前のシステムと問題点

ここで開発を行っている分散並列型動的解析システムは、SPACE に組み込まれている動的解析システムをベースにしており、大規模空間構造物を対象として幾何学的非線形性を考慮した弾塑性動的解析を実行する。数値解析手法としてはニューマークβ法を修正し、新たに提案した反復解法を用いている。非線形解析は一般に計算コストが非常にかかるので、改良前のシステムでは解析中で部材相互に依存しない部分を並列化することで計算コストの短縮を試み、その効果を確認している。

解析中の各種応力・変位状態等を表示する視覚化処理は、画面に表示を行うまでに多くの処理を必要とし、PC に高負荷を与える。そこで、並列処理システムを 3 つのサブシステム、表示機能を有して解析の制御と視覚表示を行うモニター、解析処理全体を統合・管理するマスター、並列処理部分を分担するスレーブに分け、負荷を分散する。ただし、図 2-14 に示すように、そこではマスター－スレーブ間における通信に加え、マスター－モニター間に視覚化処理のための通信処理が発生する。そのため、視覚化処理による解析処理への影響を避けるため、視覚化情報の通信をマルチスレッド（バックグラウンド）で実行する構成としている。

マルチスレッドとは複数のスレッドを並行動作させ、処理の効率化を図るもの

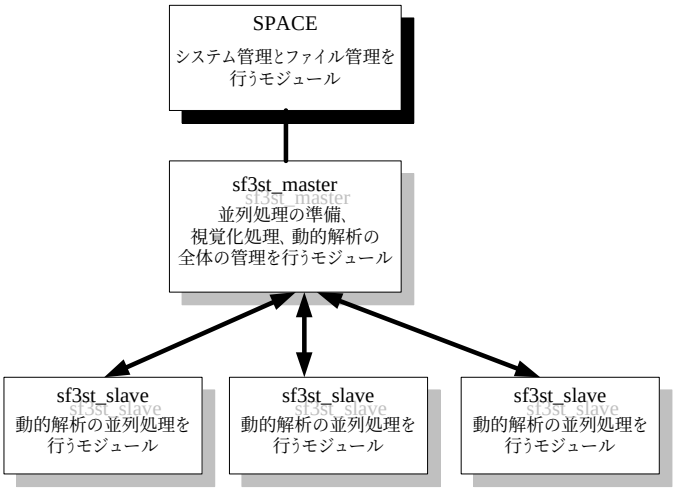


**図 2-13　改良前の並列動的解析システムの概観**

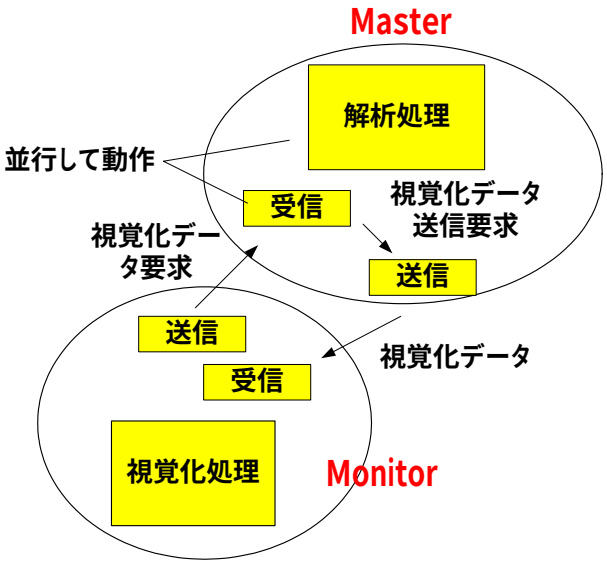


**図 2-14　視覚化処理の概観**

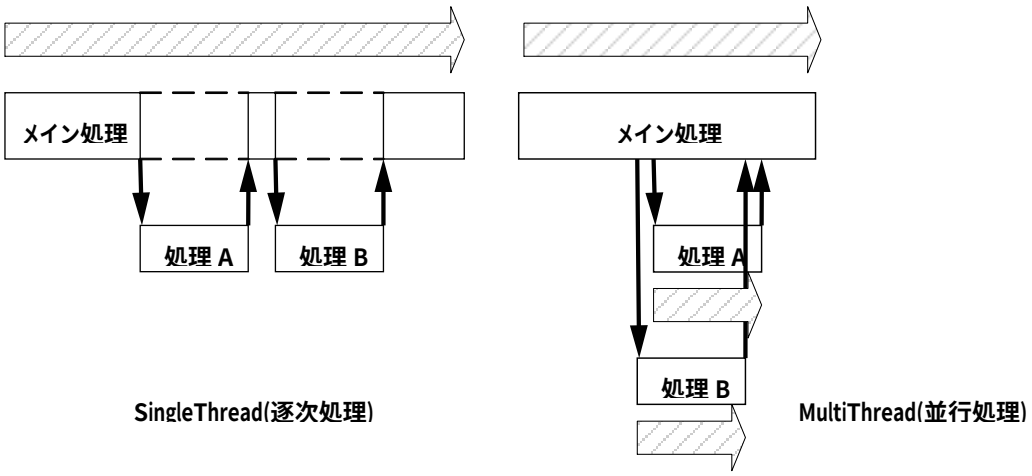


**図 2-15 逐次処理と並行処理**

である。そのイメージを図 2-15 に示す。一般に行われている処理は逐次処理と呼ばれるもので、単一の処理がコードに従って順に実行される。途中で行われる様々な処理は順番に実行され、例えば関数を呼び出した場合は関数の終了まで呼び出し側は待機し、その後次の処理を実行していく。一方、並行処理は同時に複数の処理が可能であり、メインの処理系は別の処理 A の実行を依頼した後は、その処理 A の終了を待たずに他の処理に移ることができる。そのため、並列処理では待機する時間が短縮できるので結果として全体の処理時間を短縮することが可能である。なお、スレッドとはプログラムの実行の最小単位であり、このスレッドを複数同時実行することにより前述の並列処理を行うことが可能である。

改良前では、マスターとモニターを別の PC で実行させているが、これは前述の並列処理の考えをさらに推し進めたもので、別の CPU で同時実行させることにより、視覚化による CPU への負荷が、解析処理から完全に分離され、軽減されることになる。

この視覚化処理を組み込んだ並列処理システムにおいて並列化の効果の確認を行なったが、残念ながら、視覚化処理を他の PC で行うと、並列化効率が低下し(図 2-16 参照)、解析処理も不安定となるという結果となった。

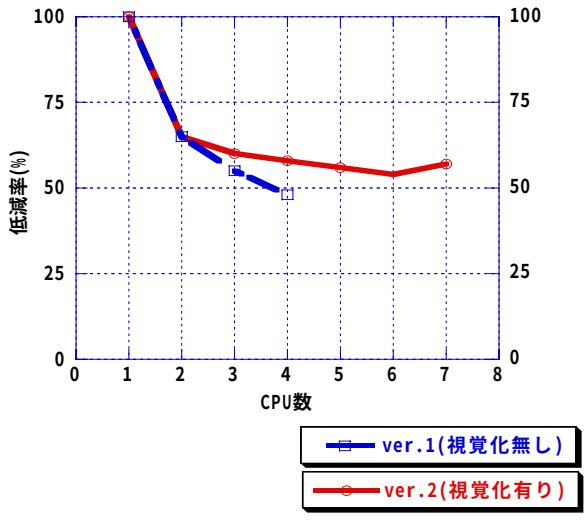


**図 3-16　過去の研究における並列効率の比較**

## 2.7 並列効率の改善案

### 2.7.1 モニターの改善

改良前のシステムでは視覚化処理を独立させたことが、逆に並列化効率を低下させる結果となってしまった。本節ではこの原因を探り、その改善案を検討する。

並列処理を実行するには、各 PC で共有情報が必要である。しかしながら、異なった PC 間では、データを完全に共有することはできない。データは主にメモリに確保されているが、一般の PC では１つのマシンに１つの CPU しか搭載されておらず、メモリはマシン間で独立して存在する。そのため、他のマシンから直接メモリにアクセスすることはできない。

別々のマシンで並列処理を行わせるためには、マシン間でのデータの共有が不可欠である。完全に独立した処理であればその必要はないが、共同で作業をしたり、解析を実行する場合は互いに情報を交換する必要があり、また解析経過をリアルタイムに表示したい場合は解析結果をモニターで参照する必要がある。しかし複数のマシン間では共通の保存領域が存在しないので、マシン A のメモリからマシン B のメモリへデータのコピーが必要となる。この場合、他の PC が管理しているメモリ上のデータを直接アクセスすることはできず、データの獲得は、一般に LAN と呼ばれるネットワークケーブルを利用した通信を用いることになる。ところが、この LAN による転送速度は図 2-17 に示すように内部デバイス間の転送速度と大きな開きがあり、例えば、コンピュータ間の外部通信速度(一般的には 100Mb/s=12.5MB/s)は、コンピュータ内のデータの内部転送速度(メモリ転送速度:1.06－3.2GB/s(1GB は 1,000MB))と比較すると 8.5 倍－25.6 倍となる。

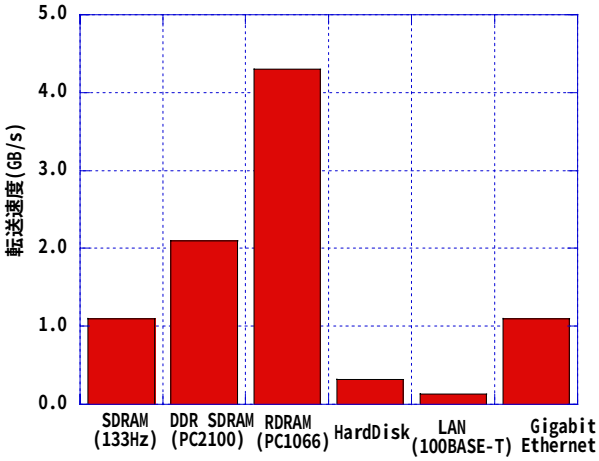


**図 2-17　転送速度の比較**

さらに、複数マシン間におけるデータ転送では、回線自体の転送時間に加え、転送相手の指定や転送データの指定など、転送のための準備時間が必要となる。この転送準備時間（通信オーバーヘッドと呼ばれる）は通信処理の効率化を考える上で大きな障害となる場合がある。表 2-4 は通信時間と通信回数の関係を表す。ケース 1 は 100 バイトのデータを 1 万回送信し、ケース 2 は 10K バイトのデータを 100 回送信している。同サイズのデータを転送する場合でも、通信回数を増やすと通信時間が飛躍

的に増大する。小さなデータ量でも、必ず通信オーバーヘッドが必要となり、上記の結果となって現れる。さらにオーバーヘッドはそれぞれのマシンで発生し、その時間は全体の処理時間に直接影響する。このオーバーヘッドを削減することは並列処理の効果を高める上で重要となる。

前節で示したように視覚化処理を独立した PC で実行するとき、システムが不安定になったり、並列処理の効率化があがらなくなる。これらの原因は、モニターとマスター間の通信時間と通信オーバーヘッドの影響であると思われる。そこで、通信量とオーバーヘッドをでき得る限り減らすために、改良システムとして視覚化機能をマスターに直接組み込むことにする。

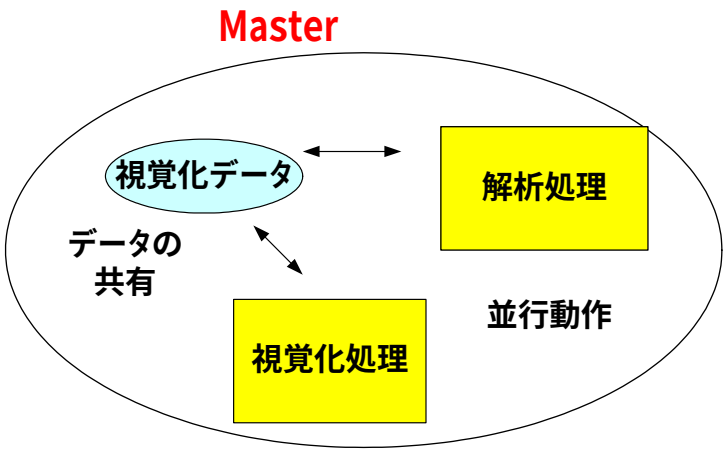


**図 2-18　視覚化機能を組み込んだマスターの構成**

視覚化処理をマスターと統合することで、視覚化する際のデータ通信が不要になる。ただし、描画の際の CPU への負荷については描画回数を最小限に抑えることにより負荷を抑制する。負荷分散の考え方から視覚化処理と解析処理を分離していたが、通信オーバーヘッドなどを考慮すると、統合した方が効率化が見込まれる。統合化にあたってはモニターをベースにし、モニターにマスター側の解析処理を追加することで、新たなマスターとしている。(図 2-18)。

**表 2-4　大きさと通信回数の異なるデータを転送したときの時間**

| 転送方式 | ケース１ | ケース２ |
|---|---|---|
| ローカル | 0.34 秒 | 0.05 秒 |
| １対１通信 | 4.68 秒 | 0.23 秒 |
| 集団通信 | 5.65 秒 | 0.46 秒 |

### 2.7.2 不安定性の改善

ここで開発を行っている分散並列型処理システムは、メッセージ通信方式の通信ライブラリーを用いて、プロセス間でデータを送受信し、データの共有化を図っている。改良前は WMPI を使用し、改良後は独自に開発した MPI_c ライブラリーを用いている。

広く利用されている通信ライブラリーとして、MPI(Message-Passing Interface)と PVM(Parallel Virtual Macine)がある。この２つのライブ

ラリーは、通信機能、通信性能、移植性などで細かな違いはあるが、基本的な概念は同じである。開発当初の分散並列処理システムでは、Windows で動作し、GUI 環境でも動作する MPI の実装ライブラリーWMPI ver1.2 を採用していたが､マルチスレッド化に十分に対応していなかったため、スレッド間でデータの共有に不具合が起きたり、複数スレッドで同時に通信を行う際に異常終了を起こしてしまうことが度々あった。そこで、今回新たな改良点として、通信ライブラリーを独自に開発し、WMPI と置き換えることにより不安定性の解消を図ることとした。

独自に並列用通信ライブラリーの開発を行う場合、「ソケット」について理解する必要がある。ソケットは PC 間の接続、受け入れ、送受信などの処理を抽象化した概念であり、このソケットを用いることで、ネットワークの構造や形式の違いなどを意識せず、通信処理を行うことができる｡SPACE 分散並列型処理システムは Windows 環境で動作しており、そのため、Windows 版のソケット「winsock」を用いて、通信ライブラリーを開発している。

このソケットによる通信は非常に低レベルで、送受信を行う場合でも常に失敗の可能性がある｡通信ハードウェア上に設けられている保存領域（プロトコル・スタック）には限界があり、一定以上のデータを一度に転送できない｡大量のデータを転送するには、指定データサイズの転送が完了するまで転送位置を未転送部分まで移動させながら送受信関数を呼び続けるという手法が必要である。他にも、大量データを安定的に送信・受信するためには多くの技術が不可欠となる。

独自に開発したライブラリーMPI_c(message passing interface compact) は並列処理用通信ライブラリーであり、主要なメッセージパッシングライブラリーの規格である MPI の一部の機能を実装したものである。ライブラリーMPI_c は、並列処理に必要な初期化･終了、送受信、状態取得などの最低限の機能だけを有する。また、MPI_c は Windows 環境でのみ動作し、呼び出し側の言語は C++と Fortran である。Fortran 版の関数は、Compaq 社の Visual Fortran90 で開発された場合のみ動作する。また、並列システムを構成するコンピュータは、TCP/IP が使用可能なネットワークに接続されていなければならない。現在のところ、MPI_c はこれらの条件を満たす環境で動作確認を行っている。

## 2.8 新システムの評価

### 2.8.1 通信ライブラリーの評価

本節では、新しく開発した通信ライブラリーMPI_c に関する通信性能の評価について述べる。通信量と通信時間の関係を調べることにより、

通信オーバーヘッドの測定と通信速度の測定を行う。測定を行なった際の測定条件を表 2-5 に示す。測定にあたってはマスター、スレーブの各 1 台でスレーブからマスターへの一方向通信とする。通信量は 1、10、100、1,000、10,000 バイトとし、それぞれ 100 回通信を行った際の通信時間を 10 回ずつ測定し、その平均値をとり、さらにその 100 分の 1 の値を 1 回の通信時間とする。

以上の条件により通信時間の測定結果を図 2-19 に示す。さらに通信時間をそれぞれの通信量ごとの通信速度（Mbps＝$10^6$ ビット/sec、1 ビット= 1/8 バイト）に変換したグラフを図 2-20 に示す。同図より 1-10 バイトの少量の通信においては、通信速度が 1-5Mbps と低速であるが、100 バイト以上では 25Mbps と高速で安定した通信が可能となっていることが分かる。1-10 バイトで低速である原因は通信量に比べ通信のオーバーヘッドの影響が強いことによる。

図 2-19 のグラフを常用対数表記にしたものを図 2-21 に示す。これにより 1-10 バイトにおいて通信速度が 10 マイクロ秒（1 マイクロ秒＝$10^{-6}$ 秒）程度のオーバーヘッドが生じていることが分かり、これが通信のオーバーヘッドであると考えられる。しかし、10 マイクロ秒程度の時間は、小規模の関数を呼ぶのとほとんど変わらない時間であり、通信のオーバーヘッドは問題ないレベルである。

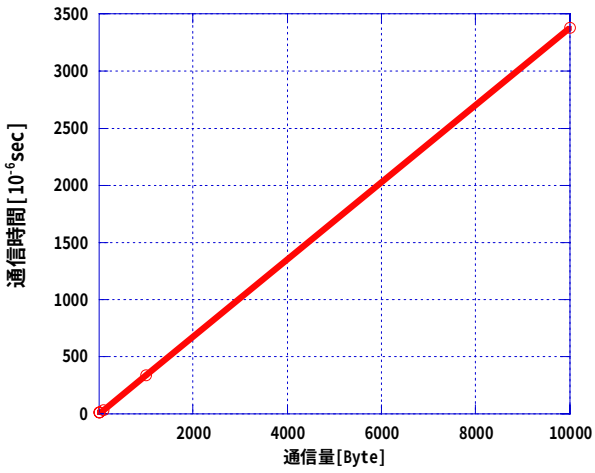


**図 2-19　通信量と通信時間の関係**

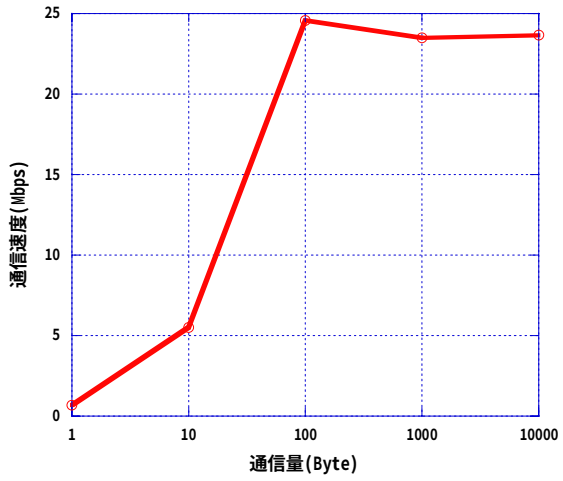


**図 2-20　通信量と通信速度の関係**

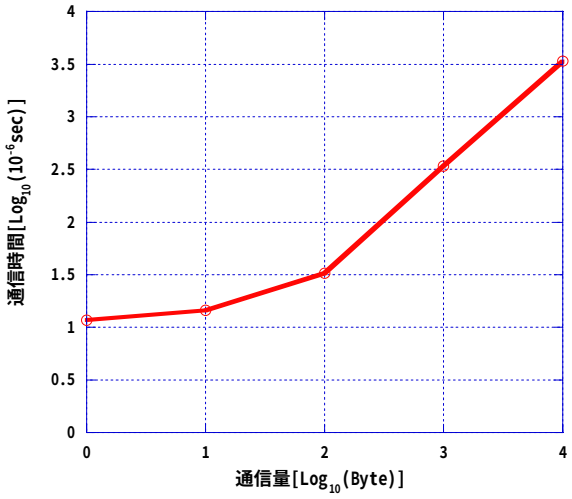


**図 2-21　通信量と通信時間の関係（常用対数表記）**

表 2-5　PC クラスターの構成

| | Master | Slave |
|---|---|---|
| CPU | Celeron1.7GHz | |
| メモリ | 256MB | |
| プロトコル | TCP/IP | |
| ネットワーク | 100Base-T | |

### 2.8.2 解析結果と考察

解析モデルは、改良前のシステムを評価した剛接合単層ラチスドーム(図 2-8 と 2-9 参照)と同一とし、弾塑性動的解析を増分時間 0.001 秒で 5 秒間行う。表 2-5 に示す構成の PC クラスターを用いて並列処理を行い、用いた PC の台数は、1 から 5 とする。

分散並列型動的解析システムは、独自に開発した通信ライブラリー MPI_c を用い、総合設計支援システム SPACE の動的解析処理部分に組み込まれている。分散並列型動的解析用ソルバーは、マスターとスレーブの二つのプログラムから構成される。また全体のシステム構成が図 2-22 に示される。

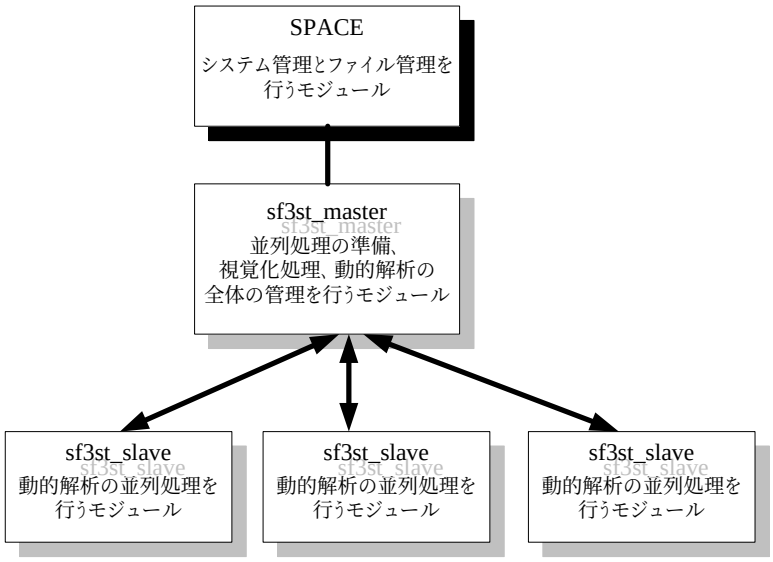


図 2-22　分散並列型動的解析システム全体の構成

上記のシステムを用いて分散並列処理を行い、その解析時間が 1 CPU に対し、どの程度低減したかを、各 CPU に対しグラフ化し、図 2-23 に示す。2 CPU で並列処理を行った場合、1 CPU の約 59%の時間で計算される。CPU の数が増えるに従い 3 CPU では約 44%、4 CPU では約 37%、5 CPU では約 33%と徐々に計算時間が短縮されており、並列処理の効果が顕著に現れていることが分かる。

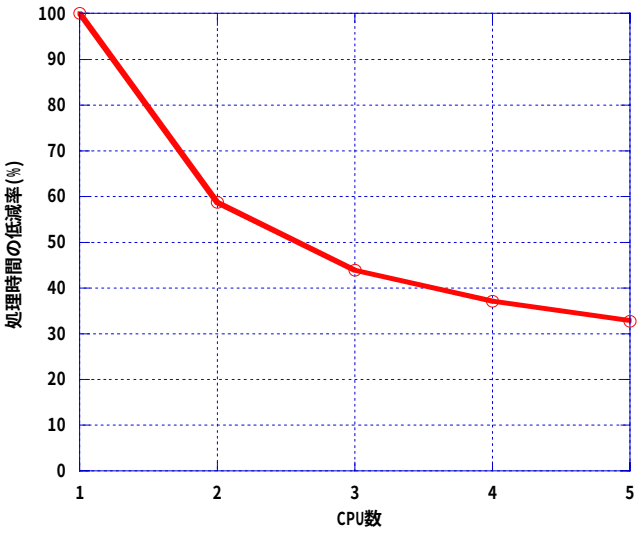


図 2-23　分散並列化による総解析時間の低減率

**表 2-6　動的解析の各処理での計算時間(秒)**

| CPU | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 予備計算 | 0.14 | 0.15 | 0.15 | 0.13 | 0.13 |
| 係数行列作成 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 |
| Newmarkβ 法 | 39.36 | 39.06 | 45.82 | 40.65 | 40.05 |
| 応力の計算 | 9.54 | 5.66 | 4.12 | 3.12 | 2.59 |
| ファイバー応力の計算 | 259.25 | 138.92 | 92.47 | 76.19 | 63.83 |
| 接線剛性作成 | 246.68 | 128.81 | 83.29 | 64.36 | 51.82 |
| 全通信時間 | 0.00 | 11.60 | 22.69 | 18.98 | 20.59 |
| 総時間 | 589.60 | 346.06 | 258.66 | 218.73 | 192.87 |

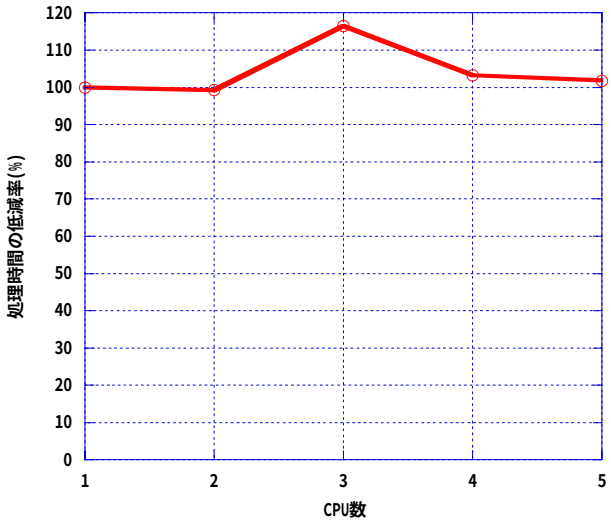


**図 2-23a 反復計算部分の低減率**

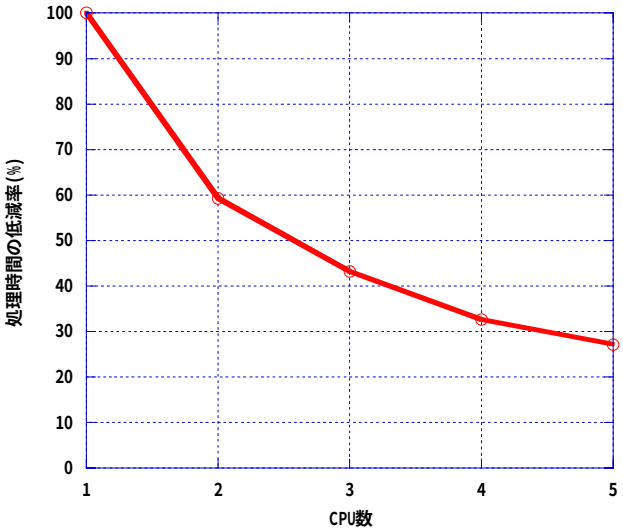


**図 2-23b 部材応力計算部分の低減率**

図 2-23a、2-23b 、2-23c は、処理全体の中で計算時間が比較的大きな割合を占める 4 つの処理について、CPU の増加に対する処理時間の推移をに示す。同図 c から分かるように、ファイバー応力の計算部分と接線剛性の計算部分は CPU 台数に応じて、2 台では約 52%、5 台では 21%と短縮されている。同じく、図 2-23b より応力の計算部分もほぼ CPU 台数に応じて約 59%から約 27%へと低減されている。

しかしながら、図 2-23a に示されるように、ニューマーク β 法による反復処理部分には、ほとんど並列化の効果が見られない。この原

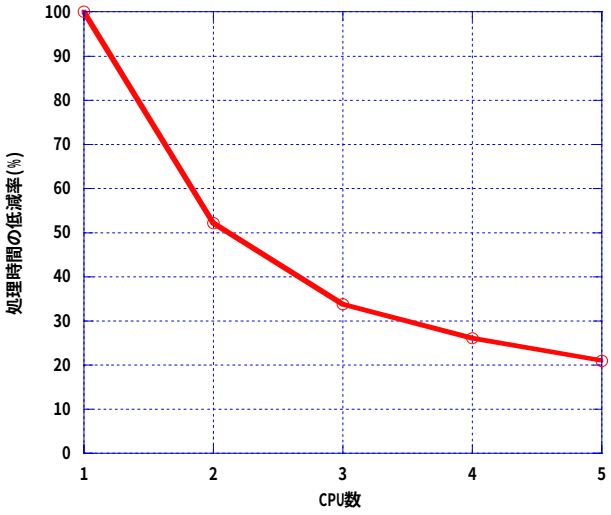


**図 2-23c ファイバー応力・接線剛性部分の低減率**

因を究明するため、ニューマークβ法の反復部分における各処理の計算時間をさらに細かく計測した。その結果を表 2-7 に示す。同表から理解できるように、並列化した非線形項$[G_i]$や右辺項作成$\{g_i\}$の計算時間は CPU 台数に応じて減少しているが、逆に、通信時間が並列化の効果と同程度に増大している。このことから、この部分の並列化を放棄するか、あるいは更なる通信時間の減少を目指す必要があることが理解できる。

通信時間の増加は、スレーブとの待ち合わせで生じている可能性がある。今後検討が必要であろう

**表 2-7 反復計算内の各処理での計算時間(秒)**

| CPU | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 右辺項作成$\{g_i\}$ | 10.42 | 5.98 | 4.42 | 3.77 | 3.31 |
| 非線形項作成$[G_i]$ | 14.16 | 7.50 | 5.06 | 4.09 | 3.38 |
| 通信時間(非線形項) | 0.00 | 9.42 | 19.85 | 15.49 | 15.68 |
| 振動方程式を解く | 13.31 | 13.45 | 13.44 | 13.80 | 13.90 |
| 通信時間(予測加速度) | 0.00 | 0.68 | 0.98 | 1.34 | 1.55 |
| 変位、速度の計算 | 0.39 | 1.18 | 1.49 | 1.92 | 2.10 |
| 通信時間(収束コード) | 0.00 | 0.22 | 0.27 | 0.31 | 0.35 |
| 反復計算[全体] | 39.36 | 39.06 | 45.82 | 40.65 | 40.05 |

### 2.8.3 並列化効率の比較

改良前後の両システムにおける並列化効率を比較したグラフを図 2-24 に示す。改良した並列処理システムでは、改良前に比較して並列化効率の低減率が良い結果を示していることが分かる。特に完全並列化である Embarrassingly parallel と比較しても良い結果が得られている。

独自に開発した並列処理ライブラリーは通信オーバーヘッドも少ないので、さらに多くの PC で並列処理を行えば、よりよい結果を出すことも可能であろう。

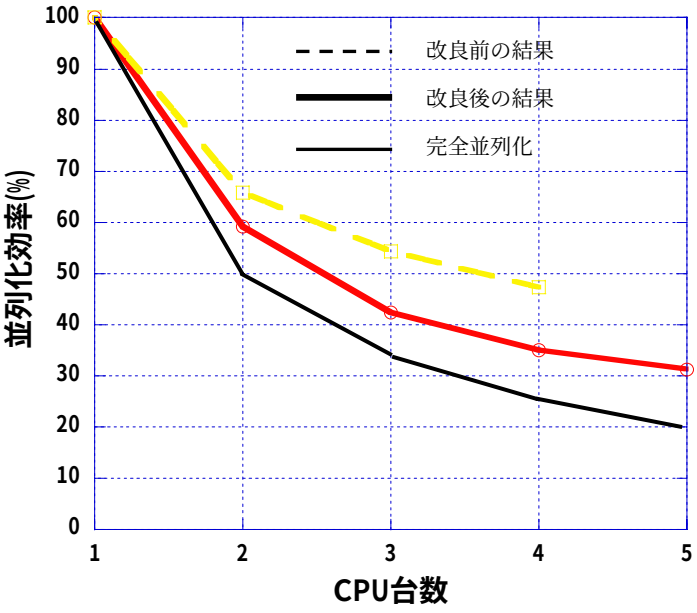


**図 2-24 改良前後での並列化効率の比較**